Vol.2015-HPC-151 No.21
2015/10/1

共有メモリ環境上でのタイルQR分解のタイルサイズチューニング

鈴木 智博[1,a)]

概要：高並列環境向きの行列分解アルゴリズムとして注目されているタイルアルゴリズムでは，問題サイズや実行環境に応じて適切なタイルサイズを選択することが非常に重要である．タイルサイズを小さくすることで，並列環境に応じた数のタスクを生成できるので負荷分散が行えるが，データサイズの低下に伴う L3 BLAS 性能の低下が起こる．つまり，環境に応じた十分な数のタスク数を供給しつつ，最大のタイルサイズを見つけることが求められる．今回，Agullo 等が提案した枝刈り探索に，生成するタスク数の基準を加えたものを使用してタイル QR 分解のタイルサイズチューニングを行った．共有メモリ環境上で行ったチューニングの結果を報告する．

Tile Size Tuning for Tile QR Decomposition on Shared Memory Systems

Tomohiro Suzuki[1,a)]

Abstract: The tile algorithms are attracting the HPC community's attention as the suitable method for a highly parallel environment. For this algorithm, it is important to select the appropriate tile size corresponding to a problem size and a run-time environment. With a smaller tile size, we can generate as many tasks as available parallel computing resources. However, it leads the performance decline of L3 BLAS routines in the implementation. The maximum tile size, which can provide the enough number of tasks is required. In order to select such tile size, we carried out the parameter tuning by using the criterion of the number of tasks in addition to the pruned search introduced by Agullo et al. In this report, we show the result of tile size tuning and the performance of the tile QR decomposition on the shared memory systems.

1. はじめに

さまざまな前処理に適用される行列分解は重要なアルゴリズムであり，科学技術計算の大規模化，高速化の要請に応えるために，高並列環境向きのアルゴリズムが求められている．我々はこれまでに QR 分解のタイルアルゴリズムをマルチコアシステムやそのクラスタシステムに実装した [12]．この実装は，OpenMP によるタスク並列プログラミングモデル，動的タスクスケジューリング [14] などの特徴を持つ．

行列分解に対するタイルアルゴリズムは，行列を小行列に分割して処理することで細粒度のタスクを多数生成できるので，高並列環境向けアルゴリズムとして研究されているが，問題サイズや実行環境に応じて適切なタイルサイズを選択することが非常に重要である．タイルサイズを小さくすることで，並列環境に応じた数のタスクを生成できるので負荷分散が行えるが，これは各タスクで扱われるデータサイズが小さくなることを意味し，これに伴って L3 BLAS 性能の低下が起こる．タイルアルゴリズムの実行パラメータであるタイルサイズに関して，環境に応じた十分な数のタスク数を供給しつつ，最大のタイルサイズを見付けることが求められる．

Agullo 等は [1] で，タイルサイズと内部ブロック幅のチューニングに関する枝刈り探索を行った．これは，行列

[1] 山梨大学大学院総合研究部工学域
Graduate School of Interdisciplinary Research, Division of Engineering, University of Yamanashi, Takeda 4-3-11, Kofu, Yamanashi 400–8511, Japan
[a)] stomo@yamanashi.ac.jp

ⓒ 2015 Information Processing Society of Japan

分解ルーチンそのものを実行してパラメータの最適値を探索するのではなく，アルゴリズム全体で支配的である 1 つのタスクのみのチューニングを行い，これより得られた最適パラメータ候補について行列分解ルーチンを実行し探索を行うものである．

今回，パラメータ探索時間をさらに短縮するために，タイルサイズに応じて変化するタスク数に着目し，タスク数の指標を加えたチューニングを行った．共有メモリ環境上で行ったタイル QR 分解のチューニングの結果を報告する．

2. タイル QR 分解

QR 分解は数値線形代数計算において重要な役割を果たす基本アルゴリズムであり，数値線形代数ライブラリ LAPACK[2] ではそのブロックアルゴリズム版が提供されている．しかし，ブロックアルゴリズムは fork-join 型の並列計算モデルに基づくため，マルチスレッド実行時には原理上必ずストールするスレッドが発生するので高並列な計算資源を高効率で利用することはできない．行列分解に対するタイルアルゴリズム [3], [4], [7] は対象とする行列を小行列（タイル）に分割し，1 または 2 タイル毎に分解，更新操作を行い，行列分解において細粒度のタスクを大量に生成することが可能である．そのため，高並列な計算資源を有効に活用できる手法として近年注目されている．

$m \times n\,(m \geq n)$ 行列 A の QR 分解は以下で定義される．

$$A = QR \tag{1}$$

ただし，Q は $m \times m$ 直交行列，R は $m \times n$ 上三角行列である．以下では，タイルサイズを $b \times b$（正方タイル）とし，行列 A は $p \times q$ 個のタイル $A_{ij} (i = 0, \ldots, p-1, j = 0, \ldots, q-1)$ から成るものとする．ただし，$p = \lceil m/b \rceil, q = \lceil n/b \rceil$ である．

行列を小行列に分割して処理するアイデアは，メモリ上に配置できない規模の行列を扱う手法として，out-of-core 分解 [6], [9], [13] で採用された．out-of-core 分解では，主メモリ上には分解または更新対象である小行列と，分解によって生成される変換行列または更新に必要な変換行列のみが配置される．それ以外の行列データはディスクに退避することで大規模行列を扱うことを可能としている．ブロックアルゴリズムのパネル分解では，変換行列を生成するために縦方向全体にわたるリダクション演算が必要であるが，パネルを縦方向にタイル分割し，最上タイルとその下のタイルとの updating QR 分解 [5] によってパネル分解を複数のタスクに分割したことで必要なデータのみ主メモリに配置する out-of-core 分解が可能となった．

タイルアルゴリズムでは大規模行列を扱うためでなく，多数の細粒度タスクを生成するために行列を小行列に分割する．後続タイル更新はタイル QR 分解において支配的であり，かつ高いデータ並列性を持つ．このため，マルチコア，メニーコアなど近年の高並列環境向きのアルゴリズムとして注目されている．操作対象以外のデータをディスクではなく他のノードのメモリ上に配置することで，out-of-core 分解の手法はクラスタ環境に容易に適用できる．タイルアルゴリズムも同様にクラスタ環境への適用は容易である．

2.1 タイル QR 分解のカーネル

タイル QR 分解は以下の 4 つの基本計算（**カーネル**）で構成される．行列の上から下を “i 方向”，左から右を “j 方向”，i，j に垂直な方向（右手系）を “k 方向” として，タイル QR 分解のカーネル実行には 3 方向の依存関係が存在する．

- `GEQRT` : 対角タイル $A_{kk}\,(k = 0, \ldots, q-1)$ のブロック QR 分解を行ない，上三角行列 R_{kk} を生成する．直交変換は，compact WY 表現 [10] により，単位下三角行列 V_{kk} とブロック上三角行列 T_{kk} に格納されるので，陽には現れない．各小行列の関係を以下に示す．

$$R_{kk} \leftarrow (I - V_{kk} T_{kk}^{\mathrm{T}} V_{kk}^{\mathrm{T}})\, A_{kk}$$

  R_{kk}，V_{kk} は A_{kk} の領域に上書きされる（ただし，V_{kk} の対角要素 1 は保存しない）．T_{kk} を保存するために別の領域が必要である．このカーネルは k 方向に依存性があり，同一タイル上の $k-1$ ステップまでのすべての更新カーネル（`SSRFB`）が終了していなければ実行できない．

- `TSQRT` : 上三角行列 $R_{kk}\,(k = 0, \ldots, q-1)$ とその下方向にあるタイル $A_{ik}\,(0 \leq k < i < p)$ の組に対してブロック QR 分解を行い，R_{kk} を更新する．直交変換は正方行列 V_{ik}，ブロック上三角行列 T_{ik} に格納される．各小行列の関係を以下に示す．

$$\begin{bmatrix} R_{kk} \\ 0 \end{bmatrix} \leftarrow \left[I - \begin{bmatrix} I \\ V_{ik} \end{bmatrix} T_{ik}^{\mathrm{T}} \begin{bmatrix} I & V_{ik}^{T} \end{bmatrix} \right] \begin{bmatrix} R_{kk} \\ A_{ik} \end{bmatrix}$$

  変換行列 V_{ik} は A_{ik} の領域に上書きされ，T_{ik} は別に確保した領域に保存される．このカーネルは i 方向，k 方向に依存性がある．つまり，A_{ik} における $k-1$ ステップまでのすべての更新カーネル（`SSRFB`）と，A_{ik} より上のタイルでの分解カーネル（`GEQRT`，`TSQRT`）が終了していなければ実行できない．

- `LARFB` : `GEQRT` によって生成された変換行列 V_{kk}，T_{kk} をその右側タイル $A_{kj}\,(0 \leq k < j < q)$ に以下のように適用する．

$$A_{kj} \leftarrow (I - V_{kk} T_{kk}^{\mathrm{T}} V_{kk}^{T})\, A_{kj}$$

  このカーネルは j 方向，k 方向に依存性がある．つまり，$k-1$ ステップまでの A_{ik} におけるすべての更新カーネル（`SSRFB`）と，V_{kk}，T_{kk} を生成する分解カー

ⓒ 2015 Information Processing Society of Japan

ネル（GEQRT）が終了していなければ実行できない．ただし，同一タイル行の LARFB は並列に実行できる．

- SSRFB：TSQRT によって生成された変換行列 V_{ik}，T_{ik} をその右側タイル A_{kj}，A_{ij} $(0 \le k < i < p, 0 \le k < j < q)$ に以下のように適用する．

$$\begin{bmatrix} A_{kj} \\ A_{ij} \end{bmatrix} \leftarrow \left[I - \begin{bmatrix} I \\ V_{ik} \end{bmatrix} T_{ik}^{\mathrm{T}} \begin{bmatrix} I & V_{ik}^{T} \end{bmatrix} \right] \begin{bmatrix} A_{kj} \\ A_{ij} \end{bmatrix}$$

このカーネルは i 方向，j 方向，k 方向に依存性がある．つまり，A_{ij} における $k-1$ ステップまでのすべての更新カーネル（SSRFB），V_{ik}，T_{ik} を生成する分解カーネル（TSQRT）と，A_{ij} より上のタイルでの更新カーネル（LARFB，SSRFB）が終了していなければ実行できない．ただし，同一タイル行の SSRFB は並列に実行できる．

上記カーネルを依存関係を考慮して実行するタイル QR 分解の擬似コードを Algorithm 1 に示す．これは Right Looking と呼ばれる静的タスクスケジューリング法に基づくものである [4]．

Algorithm 1 Tile QR factorization (Right Looking)

```
 1: for k = 0, ..., p − 1 do
 2:     (R_kk, V_kk, T_kk) ← GEQRT (A_kk);
 3:     for j = k + 1, ..., q − 1 do
 4:         (A_kj) ← LARFB (A_kj, V_kk, T_kk);
 5:     end for
 6:     for i = k + 1, ..., p − 1 do
 7:         (R_kk, V_ik, T_ik) ← TSQRT (R_kk, A_ik);
 8:         for j = k + 1, ..., q − 1 do
 9:             (A_kj, A_ij) ← SSRFB (A_kj, A_ij, V_ik, T_ik);
10:         end for
11:     end for
12: end for
```

更新カーネルの並列性から，Algorithm 1 の 3 - 5 行と 8 - 10 行の j ループは並列化可能である．例えば OpenMP による並列化を行う際は，j ループ上に並列化指示文を一行挿入するのみで，fork-join 型の並列プログラムとなる．

Algorithm 1 の第 k ステップにおいて，2 行目の GEQRT 実行後には i 方向依存が解消された TSQRT が実行可能であるが，3 - 5 行目の j ループが終了するまでは実行されない．実行可能なタスクが実行待ちとなることを極力抑えるためのスケジューリング法が動的タスクスケジューリング [14] である．これは，プログレステーブルでタスクの進捗を管理し，依存関係が解消して実行可能となったタスクをタスクキューに投入し，計算スレッドはこのキューからタスクを取り出して実行する方式である．マルチスレッド実行時には，プログレステーブルとタスクキューでアクセス競合を避けるための実装が必要となるが，これまでの我々の実験では OpenMP 実装において動的スケジューリング方式は，いくつかの静的スケジューリング方式よりも高い性能を発揮している．

3. タイルアルゴリズムのパラメータ探索

一般に科学技術計算用アプリケーションプログラムは実行時に調整可能なパラメータを持ち，パラメータが適切に設定された場合とそうでない場合では性能が大きく異なる．タイル単体の処理にブロック化を適用したタイルアルゴリズムでは，タイルサイズ b と内部ブロック幅 s が主要な調整可能パラメータである．アルゴリズム全体で支配的であり，L3 BLAS ルーチンが主要演算である更新カーネル（タイル QR 分解では SSRFB）はタイルサイズ b が大きいほど高い性能を発揮する．しかし，タイルサイズが大きくなるとタイルアルゴリズムの総タスク数は少なくなり並列計算資源に負荷不均衡が生ずる．使用するノード数やコア数などの計算資源に対して十分なタスク数を供給しつつ，可能な限り大きなタイルサイズを選択することが求められる．

Householder QR 分解は，ブロック化によって後続行列更新に高効率な L3 BLAS の適用が可能となり高速化されるが，ブロック幅分の Householder 変換を結合した Compact WY 表現の変換行列を生成するために計算量は 25% 増加する [4]．Householder QR 分解に基づくタイル QR 分解では，タイルサイズ b に応じて内部ブロック幅 s の最適値が変化するので，適切な (b, s) の組を見付けることが求められる．

Agullo 等は [1] で PLASMA ライブラリ [8] の Cholesky，LU，QR ルーチンについて，タイルサイズと内部ブロック幅のチューニングに対する枝刈り探索 (pruned search) を行った．PLASMA はタイルアルゴリズムに基づく数値線形代数サブプログラムを提供するライブラリである．以下では Agullo 等による枝刈り探索による (b, s) の探索法について述べる．

3.1 (b, s) の枝刈り探索

まず，以下を仮定する．

(1) タイルサイズ b の探索区間は $[b_s, b_e]$

(2) 内部ブロック幅 s は b の約数 $(b\%s = 0)$

上記の仮定の下で，各分解ルーチンにおいて支配的である更新カーネル単体のチューニングを行う．Cholesky，LU，QR の各ルーチンにおけるチューニング対象となる更新カーネルは，それぞれ，GEMM，SSSSM，SSRFB である．更新カーネル単体の実行時間は非常に短いので，b，s 二次元のパラメータ空間の網羅的な探索は比較的短時間で行える．ここで，タイルアルゴリズムにおけるカーネルは 1 スレッド = 1 コアで実行されることに注意する．区間 $[b_s, b_e]$ において最も高い性能を発揮するパラメータの組 (b, s) を複数個選び，最適パラメータの候補とする．想定する行列サイズの範囲でこの組についてマルチスレッドで行列分解ルーチンを実行して，各行列サイズ，実行コア数に応じた

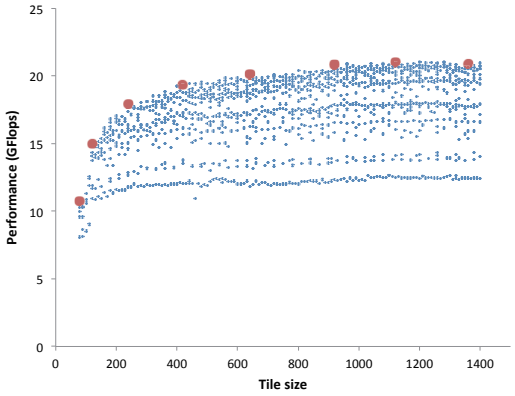

図 1 更新カーネル (SSRFB) の性能
Fig. 1 Performance of the update kernel SSRFB

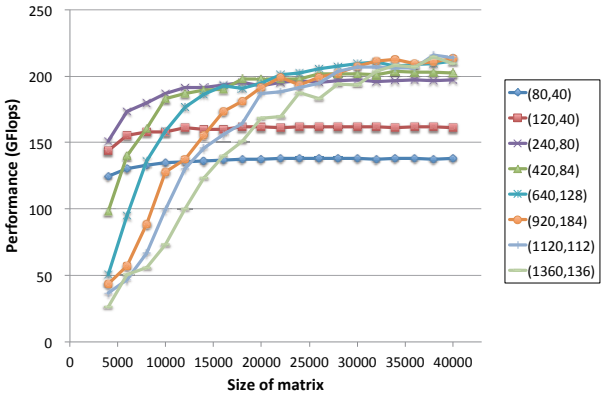

図 2 dgeqrf の性能
Fig. 2 Performance of dgeqrf

最速のパラメータの組を選ぶ．

以上が，Agullo 等の PLASMA ルーチンの枝刈り探索である．更新カーネルのチューニングから得られた最適パラメータ候補の組についてのみ行列分解ルーチンを実行する枝刈り探索と，分解ルーチンそのものを実行しながらしらみつぶしにパラメータ探索した結果を比較すると，探索時間は大きく異なるがその性能はほとんど変わらず，彼らの実験の範囲でその速度差は最大 2% 後者の方が高速であったと報告されている．

図 1 は，Xeon E5-2640 を 2 基搭載したシステムにおいて $[b_s, b_e] = [80, 1400]$ の探索区間における PLASMA のタイル QR 分解の更新カーネル SSRFB を 1 スレッドで実行した時の速度をプロットしたものである．ここで，SSRFB の計算量は $5b^3$ としている [11]．これより，同一タイルサイズにおいて，内部ブロック幅の違いにより大きく性能が異なることが分かる．この図において，各タイルサイズで最大の性能を示すパラメータの組から 8 つを適当に選び赤点としてプロットした．これが最適パラメータの候補点 $(b,s)_{Xeon\_PLASMA} = \{(80,40), (120,40), (240,80), (420,84), (640,128), (920,184), (1120,112), (1360,136)\}$ である．

次に $(b,s)_{Xeon\_PLASMA}$ の 8 つの組について，PLASMA のタイル QR 分解ルーチン dgeqrf の性能を評価する．正方行列 $(m = n)$ に対して，行列サイズ m を 4000 から 40000 まで 2000 刻みでとり，タイル QR 分解の実行時間を測定し，計算量を $(4/3)m^3$ として速度を導出した．この結果を図 2 に示す．また，表 1 に各行列サイズにおける最適な（もっとも高速な）パラメータの組を示す．

更新カーネル SSRFB の実行時間は最長で 1 秒程度であり，1 つのプロットに対し 3 回の試行を行った図 1 の総実行時間は 20 分弱であった．これに対して，同じく 3 回の試行を行った後にプロットした図 2 では，1 系列の実行時間はおおよそ 2 時間であり，すべての系列のデータを得るまでに約 16 時間かかった．これより，行列分解ルーチンそのものを実行して，しらみつぶしにパラメータ探索を行うことは多くの場合実用的でないことが分かる．

表 1 dgeqrf の最適パラメータ
Table 1 Optimal parameters for dgeqrf

$m(=n)$	b	s	$m(=n)$	b	s
4000	240	80	24000	640	128
6000	240	80	26000	640	128
8000	240	80	28000	640	128
10000	240	80	30000	640	128
12000	240	80	32000	920	184
14000	240	80	34000	920	184
16000	240	80	36000	920	184
18000	420	84	38000	1120	112
20000	420	84	40000	1120	112
22000	640	128			

図 1 から $(b,s)_{Xeon\_PLASMA}$ 以外の候補として，例えば $(540, 108)$ を選ぶと，これは $m = 22000$，24000 で表 1 の $(640, 128)$ よりも高速であるが，その実行時間の差は 1% 程度に過ぎない．表 1 の他のパラメータについても，近傍を探索してより高速な組が見つかっても大きな速度差はないので，最適パラメータの候補点を過度に多く取ることはコストの面で不利なことが分かる．

更新カーネルのみのチューニングで最適なパラメータの候補を $(b,s)_{Xeon\_PLASMA}$ に限定した枝刈り探索は，行列分解ルーチンそのものを実行しながら，しらみつぶしにパラメータ探索を行う場合に比べて大幅な時間短縮となるが，さらに最適パラメータの候補を減らすことで探索時間を削減することができるはずである．

4. タスク数の指標

[15] では，正方行列のタイル QR 分解のすべてのカーネルの実行回数における更新カーネル SSRFB の割合を以下のように算出した．各カーネルの実行回数と総数は以下の通り．

ⓒ 2015 Information Processing Society of Japan

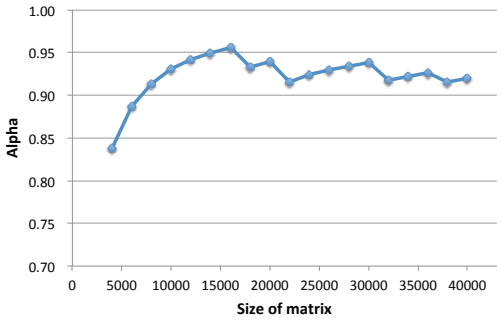

図 3 dgeqrf の最適パラメータの α 値
Fig. 3 Value of α derived from optimal parameters for dgeqrf

$$\texttt{GEQRT} \ : \ \sum_{k=0}^{p-1} 1 = p \tag{2}$$

$$\texttt{TSQRT} \ : \ \sum_{k=1}^{p-1} (p-k) = \frac{1}{2}(p-1) \tag{3}$$

$$\texttt{LARFB} \ : \ \sum_{k=1}^{p-1} (p-k) = \frac{1}{2}(p-1) \tag{4}$$

$$\texttt{SSRFB} \ : \ \sum_{k=1}^{p-1} (p-k)^2 = \frac{1}{6}p(2p^2-3p+1) \tag{5}$$

$$\text{Total} \ : \ \frac{1}{6}p(2p^2+3p+1) \tag{6}$$

ここで，全カーネル実行回数に対する SSRFB カーネルの実行回数の割合 (5)/(6) を α とする．

$$\alpha = \frac{2p^2-3p+1}{2p^2+3p+1} \tag{7}$$

$b = m$ のとき $\alpha = 0$，つまりタイル QR 分解はただ 1 つのタスク GEQRT のみを持ち，ブロック QR 分解そのものである．また，b が小さいほどタスク数は多くなり，$\lim_{b\to 0} \alpha = 1$ であるから，α が 1 に近いほどタイル QR 分解は多くのタスクを持つ．このように，更新カーネル SSRFB の実行回数の割合 $0 \leq \alpha < 1$ はタイル QR 分解のタスク数の指標として用いることが出来る．

表 1 のパラメータから算出した α 値をプロットしたものを図 3 に示す．m がある程度大きい範囲では 0.9 から 0.95 と高い値となっている．

α の範囲を $[\alpha_{\min}, \alpha_{\max}]$ とすると，(7) より p の範囲 $[p_{\min}, p_{\max}]$ が得られ，これより行列サイズ m に応じて，最適値を含むと思われるタイルサイズの探索範囲 $[b^{\alpha}_{\min}, b^{\alpha}_{\max}]$ が得られる．図 2 では，すべての行列サイズに対して，すべてのパラメータ候補 $(b,s)_{Xeon\_PLASMA}$ について QR 分解ルーチンを実行していたのに対し，各行列サイズ m に対して $[b^{\alpha}_{\min}, b^{\alpha}_{\max}]$ の範囲にあるパラメータ候補点のみを実行・評価すればよい．これによりパラメータ探索の時間を削減できる．

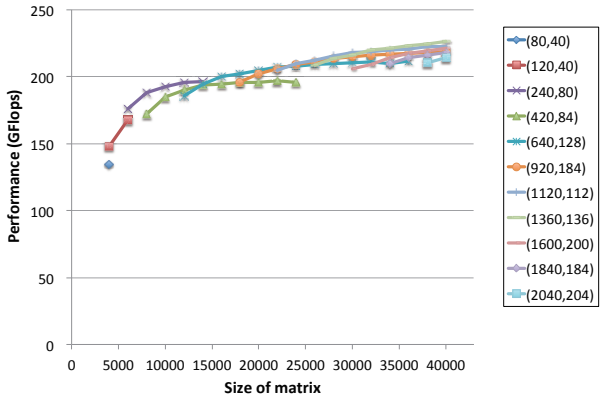

図 4 DS の性能
Fig. 4 Performance of DS

5. 実験

ここでは，我々の動的タスクスケジューリング方式の共有メモリ環境向けタイル QR 分解ルーチン [14]（以降，DS と呼ぶ）に対して最適パラメータの枝刈り探索を行う．この際 $[\alpha_{\min}, \alpha_{\max}]$ を設定して最適パラメータ候補点数を減らす．カーネルは PLASMA ライブラリのものを使用するので，最適パラメータの候補は $(b,s)_{Xeon\_PLASMA}$ をそのまま用いる．

前節の結果から $[\alpha_{\min}, \alpha_{\max}] = [0.85, 0.95]$ とした．ここで，図 3 より，$m = 4000$ と $m = 16000$ の最適タイルサイズから導出される α 値は，$[0.85, 0.95]$ の範囲から外れていることを付記する．

まず，$[\alpha_{\min}, \alpha_{\max}] = [0.85, 0.95]$ より $[p_{\min}, p_{\max}] = [18.47, 58.49]$ を得る．これより、行列サイズ m に応じて，α 値が $[\alpha_{\min}, \alpha_{\max}]$ となるようなタイルサイズ b の探索範囲 $[b^{\alpha}_s, b^{\alpha}_e]$ が得られる．行列サイズ $m = 4000, \ldots, 40000$ に対する $[b^{\alpha}_s, b^{\alpha}_e]$ を表 2 に示す．

表 2 から，$[\alpha_{\min}, \alpha_{\max}]$ を設定して導出したタイルサイズの探索範囲は 69 から 2166 であり，3.1 節で設定したタイルサイズの探索範囲 $[b_s, b_e] = [80, 1400]$ よりも広くなった．これより，更新カーネルのパラメータ探索をタイルサイズが大きい側に拡大し，最適パラメータの候補数を増やした．追加実験より，新たな候補として $(b,s)_{Xeon\_PLASMA'} = \{(1600, 200), (1840, 184), (2040, 204)\}$ を $(b,s)_{Xeon\_PLASMA}$ に追加した 11 の組の候補点について DS の性能評価を行う．

表 2 の $(b,s)^{\alpha}$ 列は，$[b^{\alpha}_s, b^{\alpha}_e]$ 内にある最適パラメータの候補点である．前述のとおり，1 つの行列サイズに対して 11 組の候補すべてについて DS を実行するのではなく，$(b,s)^{\alpha}$ の 2 つから 6 つの候補について実行すればよい．得られた DS の最適パラメータを表 3 に示す．

図 2 のように，想定する 4000 から 40000 まで 2000 刻みの行列サイズに対して，11 組のすべての最適パラメータ

ⓒ 2015 Information Processing Society of Japan

表 2 α 値から導出したタイルサイズの探索範囲と候補点
Table 2 Tile size search range derived from α and candidate parameter sets

$m(=n)$	b_s^α	b_e^α	$(b,s)^\alpha$
4000	69	217	(80,40), (120,40)
6000	103	325	(120,40), (240,80)
8000	137	434	(240,80), (420,84)
10000	171	542	(240,80), (420,84)
12000	206	650	(240,80), (420,84), (640,128)
14000	240	758	(240,80), (420,84), (640,128)
16000	274	867	(420,84), (640,128)
18000	308	975	(420,84), (640,128), (920,184)
20000	342	1083	(420,84), (640,128), (920,184)
22000	377	1191	(420,84), (640,128), (920,184), (1120,112)
24000	411	1300	(420,84), (640,128), (920,184), (1120,112)
26000	445	1408	(640,128), (920,184), (1120,112), (1360,136)
28000	479	1516	(640,128), (920,184), (1120,112), (1360,136)
30000	513	1624	(640,128), (920,184), (1120,112), (1360,136), (1600,200)
32000	548	1733	(640,128), (920,184), (1120,112), (1360,136), (1600,200)
34000	582	1841	(640,128), (920,184), (1120,112), (1360,136), (1600,200), (1840,184)
36000	616	1949	(640,128), (920,184), (1120,112), (1360,136), (1600,200), (1840,184)
38000	650	2058	(920,184), (1120,112), (1360,136), (1600,200), (1840,184), (2040,204)
40000	684	2166	(920,184), (1120,112), (1360,136), (1600,200), (1840,184), (2040,204)

表 3 DS の最適パラメータ
Table 3 Optimal parameters for DS

$m(=n)$	b	s	$m(=n)$	b	s
4000	120	40	24000	1120	112
6000	240	80	26000	1120	112
8000	240	80	28000	1120	112
10000	240	80	30000	1120	112
12000	240	80	32000	1360	136
14000	240	80	34000	1360	136
16000	640	128	36000	1360	136
18000	640	128	38000	1360	136
20000	640	128	40000	1360	136
22000	640	128			

候補について DS を 3 回ずつ実行した場合の総実行時間は約 22 時間であるのに対し，$[b_s^\alpha, b_e^\alpha]$ の範囲の最適パラメータ候補についてのみ DS を 3 回ずつ実行した場合の総実行時間は約 9 時間であった．パラメータ探索時間を大幅に削減できたことが分かる．

表 1 および表 3 の最適パラメータから算出した dgeqrf と DS の α 値を図 5 に示す．これより，行列サイズ $m = 1600$ 以降では DS の α 値の方が小さい値となっており，大きなタイルサイズ，少ないタスク数でタイル QR 分解が実行されていることが分かる．最適パラメータにおける dgeqrf と DS の性能を図 6 に示す．SSRFB の性能はタイルサイズが大きい方が高いので，全般的に α 値が小さい DS の方がわずかに高速であることが分かる．行列サイズが小さい範囲で，同一タイルサイズでも速度に若干に違いが見られるのはタスクスケジューリング方式の違いによるものと考えられる．dgeqrf は static pipeline と呼ばれる静的スケジューリング法 [7] で実行される．

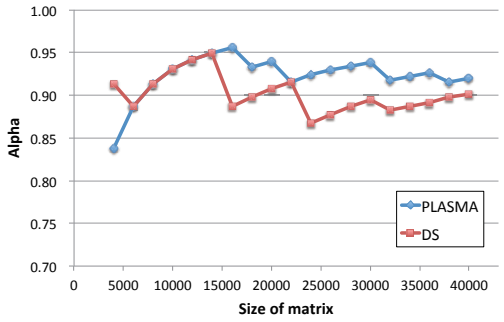

図 5 dgeqrf と DS の最適パラメータの α 値
Fig. 5 Value of α derived from optimal parameters for dgeqrf and DS

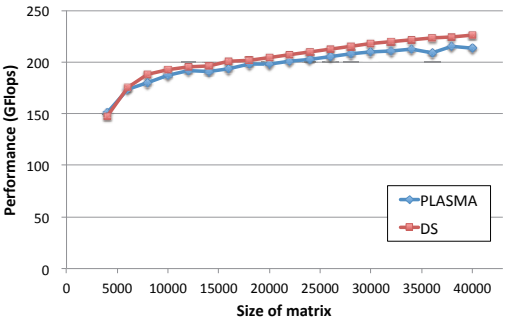

図 6 最適パラメータにおける 2 つのタイル QR 分解ルーチンの性能
Fig. 6 Performance comparison of two tile QR decomposition routines with optimal parameters

6. おわりに

行列分解に対するタイルアルゴリズムは細粒度のタスクを多く生成することで，高い並列性を持つ最近のマルチコア, メニーコアアーキテクチャの性能を発揮できるアルゴリズムとして注目されている．このタイルアルゴリズムは調整可能なパラメータであるタイルサイズ b により，その性能が大きく変化する．また，内部アルゴリズムをブロック化している場合は，タイルサイズに応じて最適なブロックサイズ s も変化する．

今回，Agullo 等の枝刈り探索法により最適なパラメー

ⓒ 2015 Information Processing Society of Japan

タの組 (b, s) が得られることを確認した．また，生成されるタスク数に関する制約を加える事で，探索する最適パラメータ候補の数をさらに減らせることを確認した．これにより，パラメータ探索時間を大幅に削減することができた．

今後は，クラスタ環境でのパラメータ探索について同様な手法で探索時間の削減が可能であるかを確認することが必要である．

謝辞 本研究は JSPS 科研費 26400197 の助成を受けた．

参考文献

[1] Agullo, E., Hadri, B., Ltaief, H. and Dongarra, J.: Comparative study of one-sided factorizations with multiple software packages on multi-core hardware, In Proceedings of SC' 09: International Conference for High Performance Computing, Networking, Storage and Analysis (2009).

[2] Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Croz, J. D., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S. and Sorense, D.: *LAPACK's user's guide, 3rd. edition*, SIAM, Philadelphia (1999).

[3] Buttari, A., Langou, J., Kurzak, J. and Dongarra, J.: Parallel tiled QR factorization for multicore architectures, *Concurrency and Computation: Practice and Experience*, Vol. 20, No. 13, pp. 1573 – 1590 (2008).

[4] Buttari, A., Langou, J., Kurzak, J. and Dongarra, J. J.: A class of parallel tiled linear algebra algorithms for multicore architectures, *Parallel Computing*, Vol. 35, pp. 38 – 53 (2009).

[5] Golub, G. H. and Loan, C. F. V.: *Matrix Computations (3rd Ed.)*, Johns Hopkins University Press (1996).

[6] Gunter, B. C. and van de Geijn, R. A.: Parallel out-of-core computation and updating of the QR factorization, *ACM Transactions on Mathematical Software*, Vol. 31, No. 1, pp. 60 – 78 (2005).

[7] Kurzak, J. and Dongarra, J. J.: QR Factorization for the CELL Processor, *Scientific Programming, Special Issue: High Performance Computing with the Cell Broadband Engine*, Vol. 17, No. 1-2, pp. 31 – 42 (2009).

[8] PLASMA: http://icl.cs.utk.edu/plasma/ (2015).

[9] Reiley, W. C. and van: POOCLAPACK: Parallel Out-of-Core Linear Algebra Package, Technical report, Tech. Rep. CS-TR-99-33, Department of Computer Sciences, The University of Texas at Austin (1999).

[10] Schreiber, R. and Loan, C. V.: A storage-efficient WY representation for products of Householder transformations, *SIAM J. Sci. Statist. Comput.*, Vol. 10, No. 1, pp. 52 – 57 (1989).

[11] Song, F., Ltaief, H., Hadri, B. and Dongarra, J.: Scalable Tile Communication-Avoiding QR Factorization on Multicore Cluster Systems, *2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '10, IEEE Computer Society, pp. 1–11 (2010).

[12] Suzuki, T. and Miyashita, H.: OpenMP/MPI implementation of tile QR factorization on T2K open supercomputer, *Proceedings of IEEE 7th International Symposium on Embedded Multicore/Many-core SoCs (MCSoC-13), Special Session on Auto-Tuning for Multicore and GPU (ATMG)* (2013).

[13] Toledo, S. and Gustavson, F. G.: The Design and Implementation of SOLAR, a Portable Library for Scalable Out-of-Core Linear Algebra Computations, *WORKSHOP ON I/O IN PARALLEL AND DISTRIBUTED SYSTEMS*, ACM, pp. 28 – 40 (1996).

[14] 鈴木智博：タイルアルゴリズムのための動的スケジューラの OpenMP 実装，情報処理学会研究報告ハイパフォーマンスコンピューティング (HPC)，HPC-139, No. 13, pp. 1 – 6 (2013).

[15] 鈴木智博：クラスタシステム向けタイル QR 分解のタイルサイズチューニング，情報処理学会研究報告ハイパフォーマンスコンピューティング (HPC)，HPC-146, No. 15, pp. 1 – 6 (2014).

ⓒ 2015 Information Processing Society of Japan